Recent Innovations in Mechatronics (RIiM) Vol. 5. (2018). No. SI.
DOI: 10.17667/riim.2018.si/17.

Robosztus objektumfelismerés mobil robotokhoz, Kinect szenzor segítségével

Szabó András Gergő
Mechatronikai Tanszék
Debreceni Egyetem, Műszaki Kar
Debrecen, Magyarország
szabo.andras.gergo@gmail.com

Dr. Szemes Péter Tamás
Mechatronikai Tanszék
Debreceni Egyetem, Műszaki Kar
Debrecen, Magyarország
szemespeter@eng.unideb.hu

Dr. habil. Husi Géza
Mechatronikai Tanszék
Debreceni Egyetem, Műszaki Kar
Debrecen, Magyarország
husigeza@eng.unideb.hu

***Absztrakt*— Geometriai jellemzőkkel megadott objektumdefiníció alapú objektum felismerő rendszer LabVIEW-ban kódolva, ahol a felvételeket Kintect szenzorral oldottuk meg.**

Kulcsszavak—objektum felismerés, ajtó detektálás, objektum definíció, LabView, Vision Assistant

BEVEZETŐ

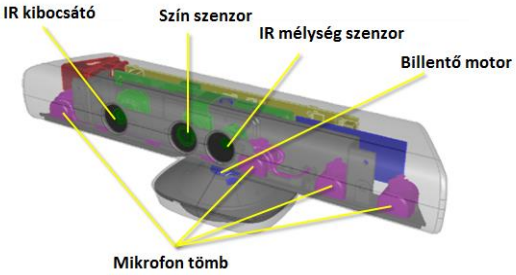
Kutatásunkban egy olyan hibrid algoritmus létrehozásának lehetőségeit vizsgáltuk, amely egy célorientált felismerő rendszer, és amelyben az egyes objektumok a képjellemzők relatív helyzete alapján vannak definiálva. Ilyen módon modelleztük egy beltéri nyitott ajtót, de az analógiát alkalmazva más objektumot is felismerhetnénk, a használt Kinect 1414 pontossági korlátait figyelembe véve. Vizsgálatokat végeztünk arról, hogy egy ilyen felismerő rendszerben a feldolgozandó képen az ajtó mely jellemzői jól felismerhetők, a zajokra kevésbé érzékeny, és mely kinyert geometriai jellemző mennyire tér el a valóstól.

I. ROBOSZTUS OBJEKTUMFELISMERÉS DEFINIÁLÁSA

A fejlesztett algoritmus olyan jellemzők alapján ismeri fel az ajtókat, amelyek változó fényviszonyok, és látószög mellett is jól felismerhetők, nem kell egy objektumosztály minden példányát egyesével betanítani. Így képes bármely beltéri nyitott ajtó felismerésére, ez az előnye más modell alapú felismerő rendszerekkel szemben., jó objektumdefiníciókat kell megfogalmaznunk, és minden tárgy osztályhoz külön kell meghatároznunk.

II. A MÉRŐRENDSZER RÉSZEI ÉS A KÉPFELDOLGOZÁSHOZ HASZNÁLT SZENZOR

Microsoft Kinect 1414-es modell: A Kinect kamerát egy új fajta parancs beviteli perifériaként hozta létre a Microsoft az XBOX 360 konzolhoz, amelynek első megjelenése 2010-re tehető. Az eszköz tartalmaz egy színes kamerát, valamint egy infravörös fényforrást és egy infravörös kamerát. Az infravörös adó-vevő párossal egy mélységkép áll rendelkezésünkre, amely mérési tartománya 800-8000 mm között található.
Dell Latitude E6440 Laptop, A felismerő algoritmus létrehozására LabVIEW 2016 szoftvert használtuk



1. ábra A kinect felépítése

1. táblázat A Kinect 1414 legfontosabb tulajdonságai

Mélységkép felbontás	*640x480*
Színes kép felbontás	*640x480*
Mélység méréstartomány	800mm-8000mm
Függőleges látószög	43°
Vízszintes látószög	57°

III. A MÉRÉSADATOK GYŰJTÉSE

A Kinect és a számítógép közötti kommunikáció megteremtéséhez a Kinesthesia Toolkitet használtuk. Ez egy alkalmazás programozási felület (API), amely könnyen feldolgozhatóvá teszi a Kinect által küldött adatokat a LabVIEW-ban.

IV. A KINECTBŐL ÉRKEZŐ ADATOK

A Kinesthesia Toolkit Read.vi kódrészlet kimeneteként két tömböt kapunk. A mélységértékeket egy jelöletlen 16 bites értékeket tartalmazó, a színes kép pixeleinek értékeit szintén jelöletlen 24 biten tárolt számokat tartalmazó tömbben érkeznek.

V. A MÉLYSÉGADATOK VIZUALIZÁLÁSA

Létrehoztunk egy SubVI-t, amely a mélységértékeket HSV (Hue, Saturation, Value) színtérbe transzformálja, amelyet tovább alakít RGB színtérbe.

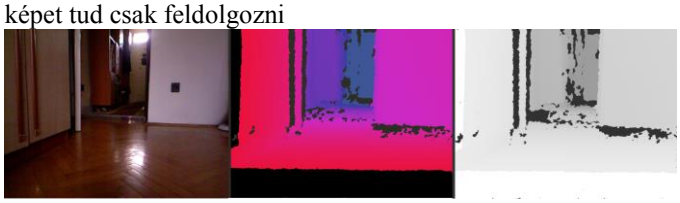
$$K = 10000 - d_{x,y_{mm}} \quad (1)$$

$$H_{d_{x,y}} = \frac{K}{Hue\ mod} \quad (2)$$

$$S_{d_{x,y}} = \frac{K}{Sat\ mod} \quad (3)$$

$$V_{d_{x,y}} = \frac{K}{Val\ mod} \quad (4)$$

ahol K egy korrekciós tényező, amely a színskála színeit meg fordítja, ezzel is növelve az értelmezhetőséget. A *Hue mod, Sat mod* és *Val mod* rendre a HSV értékek módosító értékei. A transzformációs VI segítségével kétféle színskálát alkalmaztunk a mélységképekre: (1) Színes skála a jobb vizualizációs élményért, egyszerűbb a tárgyak geometriai tulajdonságai elemezni. Képfeldolgozásra ezt a skálát nem használtuk. (2) Szürkeárnyalatos skála a képfeldolgozáshoz. Kontrasztosabbak az élek, mint a színes képen, valamint a Vision Assistant alakzatkereső funkciója szürkeárnyalatos képet tud csak feldolgozni



2. ábra Balról jobbra: színes kép, színes mélységkép, szürkeárnyalatos mélységkép

VI. A FELDOLGOZÓ ALGORITMUSOK

A. *Beltérben használatos ajtók*

Jelenleg az európai uniós jogharmonizáció óta nincs érvényes szabályozás a beltéri ajtó méretekre, ezért a már hatályon kívül lévő elterjedt MSZ beltéri egyszárnyas ajtó nyílásméreteket használom alapul: 75x210 cm,90x210 cm,100x210 cm,140x210 cm. Megállapítható, hogy a magyar háztartásokban használatos ajtók belő méretének szélessége 60-130 cm között változhat, míg a magassága közel állandó 200 cm körül található az ajtókeret vastagságától függően.

B. *Az ajtó objektum definíciója*

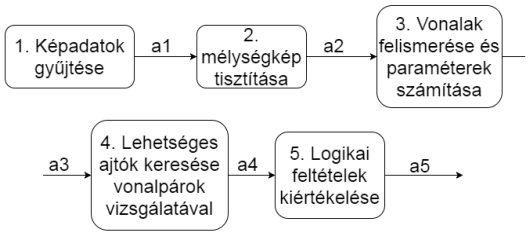
Kutatásunk csak háztartásban használatos nyitott beltéri ajtók felismerését célozza meg. Az ajtó felismeréséhez az ajtókeret éleit próbáltuk meg detektálni A Vision Assitant beépített kódjai segítségével. A vonalakat vonalpárokba rendezve vizsgáltam tovább, és az ajtó jelenlétét.

$$A = ME + MA + SZ + ÁM \quad (5)$$

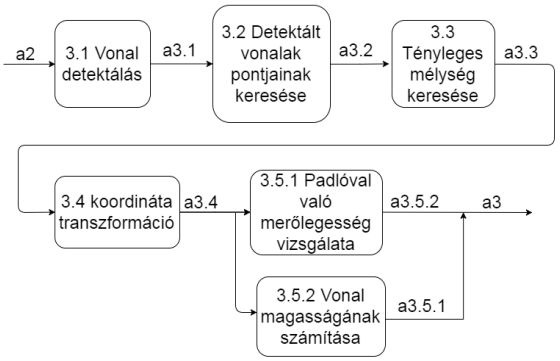
Ahol:

- ÁM: a felismert vonalak meghatározott területek átlagos mélységének különbsége megfelelő
- SZ: vonalpár szélessége nagyobb mint 500 mm és kisebb mint 1500 mm
- MA: vonalak magassága nagyobb mint 500 mm
- ME: vonalpár tagjainak θ és ϕ szögei kisebbek mint 10°

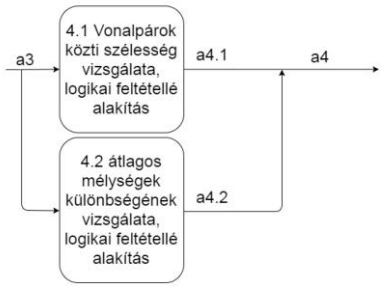
C. *A detektálás folyamata*



3. ábra A feldolgozó algoritmus blokkvázlata

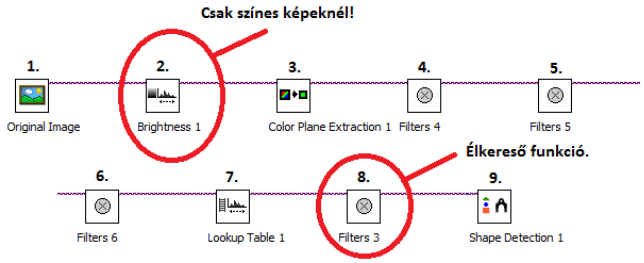


4. ábra A felismerő algoritmus 3. pontjának blokkvázlata



5. ábra A felismerő algoritmus 4. pontjának blokkvázlata

A blokkvázlat 3.1. pontjában a vonalak detektálásához a Vision Assistant-ot használtuk, és azon belül különböző szűrőket és élkereső eljárásokat. A felismert vonalak pontossága kritikus a helyes felismeréshez, ezért több módszert vizsgáltunk a megfelelő eredmény eléréséhez. Az algoritmusok közel ugyanazon elemekből állnak, eltérés az élkereső funkcióban van, valamint abban, hogy a színes képek első lépésként fényerő növelésen esnek át.
Az alkalmazott élkeresők: (1) Színes képen: vonal detektálás előzetes élkeresés nélkül, (2) Canny élkereséssel, és Differenciális élkereséssel, mélységképen: (3) Vonal detektálás előzetes élkeresés nélkül, Laplace élkereséssel, és Differenciális élkereséssel.



6. ábra Élkereső algoritmus blokkvázlata színes képen Canny élkiemeléssel

Jelmagyarázat:

1. Bejövő kép,
2. Fényerő növelés,
3. Fényerő színsík kivonása (ezt tartjuk meg),
4. Medián elmosó szűrő,
5. Részlet kiemelő szűrő,
6. Gauss elmosó szűrő,
7. Hatványozó keresési tábla,
8. Vonal detektálás.

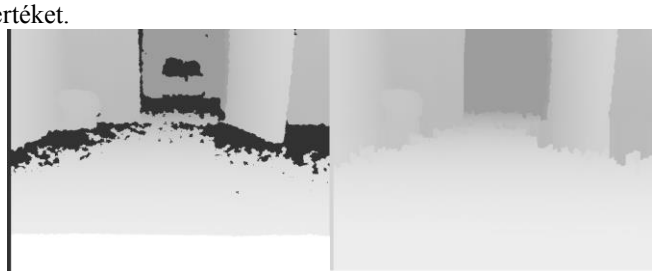
A 4, 5, 6,7 pontok szerepe az előzetes kontrasztnövelés anélkül, hogy a bemenő kép zajait is felerősíteném, ezért ezeket minden algoritmusban használtuk.

VII. AZ OBJEKTUMFELISMERŐ RENDSZER MEGÉPÍTÉSE

A blokkvázlat 3.1. blokkjának kimenete a felismert vonalak kezdő és végpontjainak koordinátái. Ezeket a vonalakat konvertáltuk 3D koordináta rendszerbe, hogy meg tudjuk vizsgálni, hogy a képen közel függőlegesnek tűnő vonalak a valóságban mekkora szöget zárnak be padlóval. A vonalak felismerése úgy történik, hogy az élkereséssel talált pontokra egyenest illeszt a szoftver, ez azonban azzal járhat, hogy a vonal különböző szakaszain eltérő mélységértékeket kapunk: vagy az ajtókeret távolságát, vagy az ajtókeret mellett az ajtón túl található tárgyak távolságát, vagy hibás adatot. Nekünk az elsőre van szükségünk.

A. A mélységkép megtisztítása

A mélységképen megjelenő fekete foltok három okból adódhatnak: A mérendő távolság a mérési tartományon kívül esik, a tükröződő, átlátszó felületeken, vagy nagy mélységkülönbségek peremén. A mélységtömbben ilyen mérések esetén 8191-es érték jelenik meg. Ezen értékek kiküszöbölésére a következő írtunk egy algoritmust, amely végighalad a kép összes pixelén, és minden helyen, ahol 8191-es értéket talál, egymás után vizsgálja a körülötte lévő elemeket egyre távolabb, amíg nem talál egy nem hibás értéket.



2. ábra Jobbra a tisztítás előtti, balra a tisztítás utáni mélységkép

B. A tényleges mélység számítása

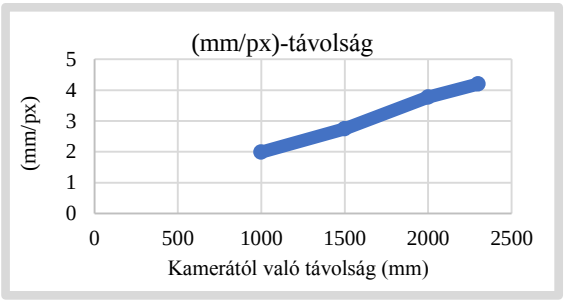
Nem elég a képre illesztett vonal, azok a pontokra is szükségem van amire az egyenest illeszti az algoritmust. Mivel a vonal detektor nem adja ki kimenetkén a vonalillesztéshez felhasznált pontokat, ezért a vonalak kezdő és végpontjai közötti területet 5 részre osztva, A Vision Assistant élkeresés funkciójával kerestük azokat. Ez után az élpontok 30 pixeles távolságában balra és jobbra, a pixelek mélységértékeinek móduszát vettük, és végül ezek lettek a pontokhoz tartozó mélységértékek.

C. Koordináta transzformáció

Ahhoz, hogy könnyen, egyszerű koordináta-geometriai számításokkal tudjam számolni a pontok távolságát, és a merőlegességet *x, y* pixelkoordinátákat és a pontokhoz tartozó mélységértékeket térbeli derékszögű koordináta rendszerbe számítottuk át.

3. ábra A térbeli koordinátarendszer tengelyei

Az átszámításhoz szükségünk volt arra, hogy hogyan változik a tényleges/pixelméret aránya a távolság függvényében. Ezen érték számításához méréseket végeztünk, mely során 6 különböző ismert méretű tárgy pixelméretét vizsgáltuk különböző távolságokon. Az objektumok valós mérete sorszám szerinti növekvő sorrendben 234 mm, 376 mm, 428mm, 806 mm, 1061mm, 1280. A pixel méret függvényét a 8. ábra mutatja.



4. ábra A (mm/px)- távolság diagram

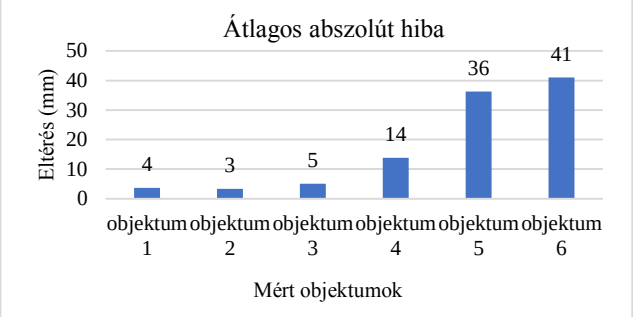
A diagram alapján látható, hogy a $\frac{mm}{\frac{px}{\Delta d}}$ érték közel konstans. Értékét átlagolás alapján 1,878-ra választottuk.
A tényleges méret számítása:

$$m_{mm} = d \times 1{,}878 \times m_{px} \qquad (6)$$

ahol:

- m_{mm}: méret milliméterben,
- d (depth): mélységérték mm-ben,
- m_{px}: méret pixelben

A képlet alapján visszaszámolt méret és a tényleges méret közötti eltéréseket a 9. ábra foglalja össze.



5. ábra A visszaállított méretek átlagos abszolút hibája

A koordináta transzformáció:

$$\begin{pmatrix} x_{mm} \\ y_{mm} \\ z_{mm} \end{pmatrix} = \begin{pmatrix} (x_{px} - 320) \times d \times 1{,}878 \\ (240 - y_{px}) \times d \times 1{,}878 \\ d \end{pmatrix} \quad (9)$$

D. Padlóval való merőlegesség vizsgálata

Meghatároztuk a vonalak pontjainak térbeli koordinátáit, a padlóra való merőlegességet a padló normálvektorának, és a vonal irányvektorának párhuzamosságának vizsgálatával határoztuk meg. Ehhez szükség volt a vonal irányvektorára, amelyet a kezdő és végpontok koordinátáival számítottunk.

$$\begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} = \begin{pmatrix} x_2 \\ y_2 \\ z_2 \end{pmatrix} - \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} \quad (10.)$$

ahol v_x, v_y, v_z az irányvektor, x_2, y_2, z_2 a vonal végpontjának, x_1, y_1, z_1, pedig a vonal kezdőpontjának koordinátái.
A padló normálvektorát az ideális

$$\bar{n} = (0,1,0) \quad (11.)$$

értékre vettem. Ahhoz, hogy a normálvektor és az irányvektor eltérését vizsgáljam a pontokat áttranszformáltuk térbeli polár koordináta rendszerbe:

$$\begin{pmatrix} r \\ \theta \\ \varphi \end{pmatrix} = \begin{pmatrix} \sqrt{x^2 + y^2 + z^2} \\ \arctan\left(\frac{y}{x}\right) \\ \frac{z}{\sqrt{x^2+y^2+z^2}} \end{pmatrix} \quad (12.)$$

6. ábra A polár koordinátarendszer paraméterei

A merőlegesség logikai kimenetet úgy határoztuk meg, hogy a vektorok θ és φ szögeinek eltérése nem lehet nagyobb 10°-nál:

$$|\theta_n - \theta_v| \leq 10°, \quad (13.)$$

$$|\varphi_n - \varphi_v| \leq 10°, \quad (14.)$$

ahol: θ_n φ_n a normálvektor-, θ_v φ_v a vonal irányvektorának koordináta tengelyekkel bezárt szögei.

E. A vonalak magasságának vizsgálata

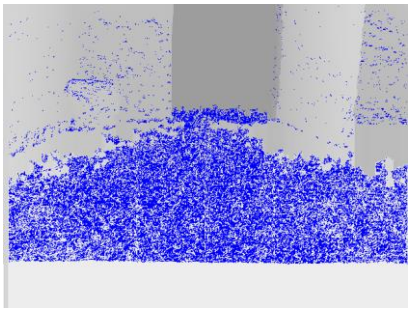
Mivel a felismert vonalak legtöbbször nem fedik teljes mértékben a valós élt, ezért a vonalakat a kezdőpontjuktól a padlóval való metszéspontjukig vizsgáltuk. Ez bizonyos esetekben hibát is okozhat, például, ha egy falon lévő képkeret élét a padlóig hosszabbítja, de ezzel a problémával azért nem foglalkoztam, mert a felismerő algoritmus többi része kiküszöböli az itt bevitt hibát, és a vésző kimenetet nem befolyásolja.
Vonal és a padló síkjának metszéspontja:

$$M = \frac{n \cdot (P_p - P_v)}{n \cdot v} \quad (15.)$$

ahol:

- M: metszéspont,
- *n*: a padló normál vektora,
- *Pp*: a padló egy pontja,
- *Pv*: a vonal egy pontja,
- *v*: a vonal normálvektora.

A padló egy pontjának meghatározásához a normálvektoros síkkivonást alkalmaztunk. A kód egy egyszerűsített síkfelismerő algoritmus, de arra elég, hogy a padló egy pontját meghatározzuk. A kék pixelek jelölik a hasonló irányú normál vektorokat, amelyek θ és φ szögeinek különbsége nem nagyobb mint 10°. A felismerő program jelen szakaszában a felismert padló pontok közül kézzel határoztuk meg a választottat. A vonal magasságának kiszámolásához egy kódot hoztam létre, amely két térbeli pont távolságát számítja:



7. ábra normálvektoros síkkivonás a padló egy pontjának meghatározásához

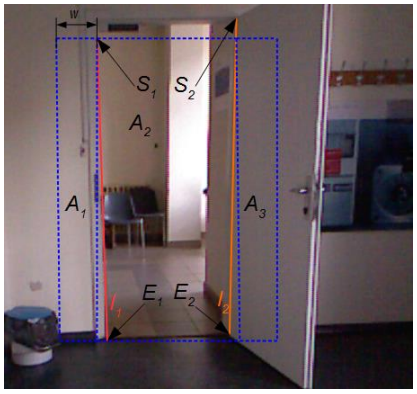
$$D = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}, \quad (16.)$$

ahol x_1, y_1, z_1 az egyik x_2, y_2, z_1 pedig a másik pont koordinátái, *D* pedig a két pont közötti térbeli távolság. A vonalak magasságát minimum 500 mm-ben állapítottuk meg, mert így akkor is képes felismerni az ajtót a program, ha a teteje nem látszik, valamint egy a program elkészítését motiváló mobil robot átférne az ajtón.

F. A vonalpárok vizsgálata

A vizsgálat során minden vonalat minden vonallal párosítottunk egyszer, majd az előző pontban leírtak szerint számítottam a két vonal közti távolságot. Az utolsó logikai feltételként a következőket határoztuk meg: Az 12. ábrán látható A_1 és A_3 területek mélységértékeinek átlaga legalább 1000 mm-rel legyen nagyobb, mint A_2 terület mélységértékeinek átlaga.

DOI: 10.17667/riim.2018.si/17.



8. ábra Az átlagos mélységértékek területei

$$A_{A_1} = \frac{\sum_{i=S_{1,x}-w}^{S_{1,x}} \sum_{l=S_{1,y}}^{E_{1,y}} d_{i,l}}{i \times l} \quad (17.)$$

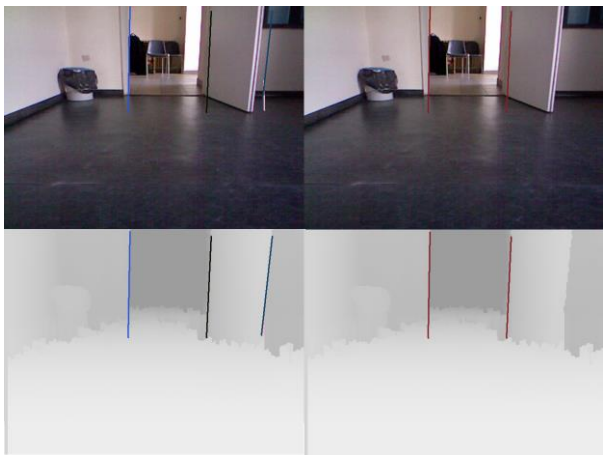
$$A_{A_2} = \frac{\sum_{i=S_{1,x}}^{S_{2,x}} \sum_{l=S_{1,y}}^{E_{1,y}} d_{i,l}}{i \times l} \quad (18.)$$

$$A_{A_3} = \frac{\sum_{i=S_{2,x}}^{S_{2,x}+w} \sum_{l=S_{1,y}}^{E_{1,y}} d_{i,l}}{i \times l} \quad (19.)$$

Ahol

- $S_{1,x}$, $S_{1,y}$ az l_1 vonal kezdőpontjának pixelkoordinátái,
- $S_{2,x}$, $S_{2,y}$ az l_2 vonal kezdőpontjának pixelkoordinátái,
- $E_{1,x}$, $E_{1,y}$ az l_1 vonal végpontjának pixelkoordinátái,
- $E_{2,x}$, $E_{2,y}$ az l_2 vonal végpontjának pixelkoordinátái,
- w a keresési szélesség.

G. *A mérések eredménye*

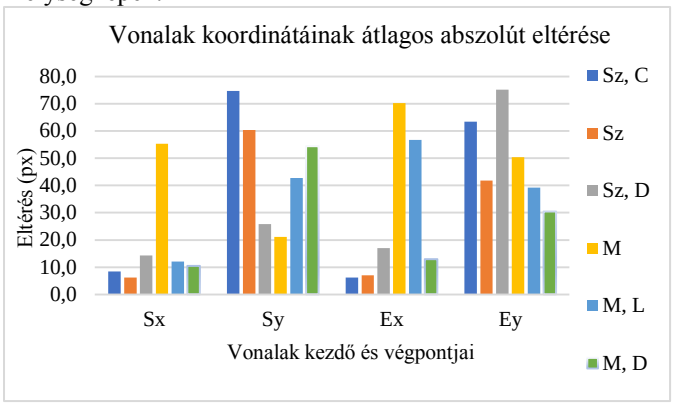


9. ábra felismert vonalak és ajtók

Felvételeket készítettünk a Debreceni Egyetem Műszaki Karán a Mechatronikai Tanszék épületében. A felvételekről kiválasztottunk 3 képet, melyekről az összes algoritmussal felismert vonal összes ki vont adatát összegyűjtöttük, amely összesen 167 vonalat jelent.

A három képből kettő az ajtó különböző oldalán van, és mindhárom eltérő magasságból és eltérő szögből készült.

A bal felső ábrán a mélységkép alapján Laplace élkereséssel felismert vonalak, alatta ugyan ezek a mélységképen, a jobb felsőn a felismert ajtó látható, a jobb alsón ugyanez a mélységképen.

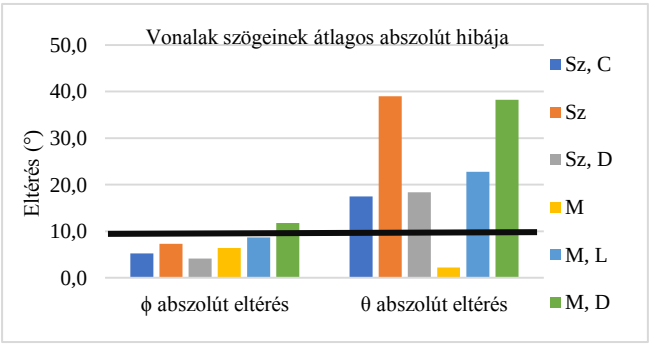


10. ábra Felismert vonalak koordinátáinak a ténylegestől való átlagos eltérése

Jelmagyarázat:

- *Sx*, *Sy*: Vonal kezdppontjának *x* és *y* koordinátái
- *Ex*, *Ey*: Vonal végpontjának *x* és *y* koordinátái
- Sz, C: Színes képen Canny élkiemelés
- Sz: Színes képen élkiemelés nélkül
- Sz, D: Színes képen differenciális élkiemelés
- M: Mélységképen élkiemelés nélkül
- M, L: Mélységképen Laplace élkiemelés
- M, D: Mélységképen differenciális élkiemelés

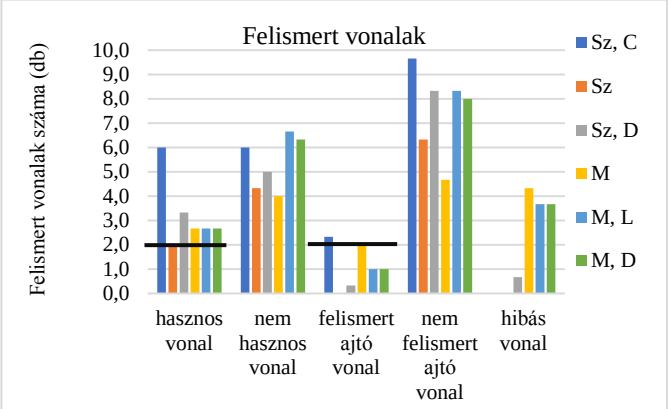
Látható, hogy az algoritmusok az ajtó élek végpontjainak felismerésében ejt a legnagyobb hibát. A hiba elsősorban az élkeresési funkció miatt van, hogy nem emeli ki teljes egészében a vonalakat. A színes képen keresett élek megtalálása a fényerőtől és megvilágítástól is függenek, melyek hatását most nem vizsgáltam. Az *y* tengely irányú hibák nagyobbak, ezért elsősorban az a módszer tekinthető jobbnak, amelyik ebben jobban teljesít, ezek a Sz,D az M, az



M, D és az M, L jelzésű módszerek.

11. ábra A felismert vonalak szögeinek eltérése a valóságostól

A θ értékek hibája azért nagyobb mert, ha az algoritmus rosszul számította a mélységet, akkor előfordulhat, hogy például a vonal alsó pontja az ajtókeret mélységét-, a vonal felső pontja pedig az ajtón túli tárgy mélységét kapja, amely azt eredményezi, hogy az él *xy* síkkal bezárt szöge nagy lesz. Ez látható az 17. ábrán a 25°-nál nagyobb θ eltéréseknél. A fekete vonal a logikai feltételek között megtalálható megengedett θ és φ eltéréseket mutatja.

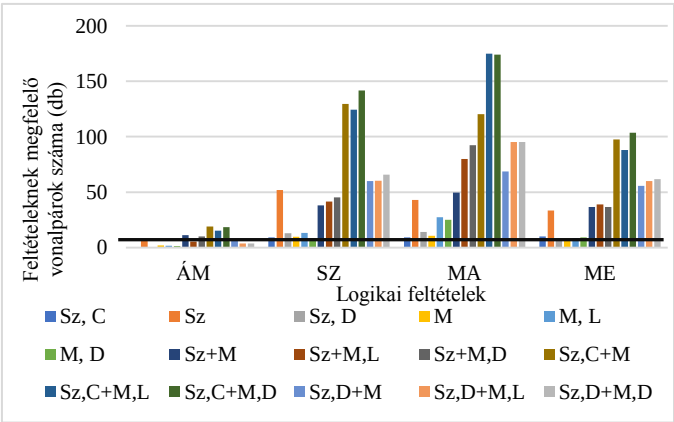


12. ábra A logikai feltételeknek megfelelő vonalak száma algoritmusonként

A jelölt vonaltípusok magyarázata:

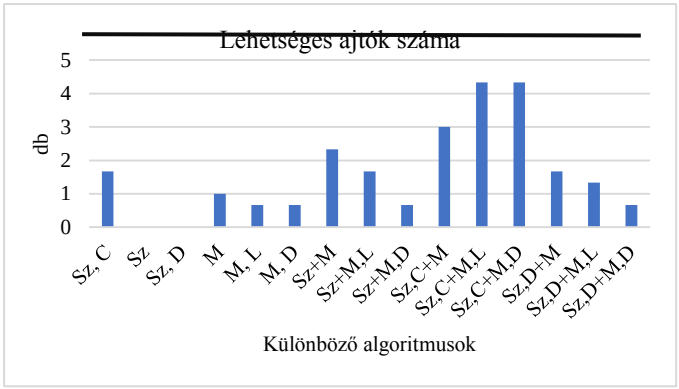
- hasznos vonal: tényleges ajtó él, amelyet megtalált a program
- nem hasznos vonal: felismert él de nem az ajtó éle,
- felismert ajtó vonal: felismert ajtó egyik éle, azaz az összes logikai feltételnek megfelel.
- nem felismert ajtó vonal: a program felismerte a vonalat, de a logikai feltételeknek nem tett eleget, ezért nem ismerte fel, mint ajtót,
- hibás vonal: rosszul számított mélységértékek miatt hibás 3D koordinátákat kapott így az él a valóságban nem létezik

A fekete vonal azt jelzi, hogy mivel minden tesztképen volt egy ajtó, ezért elvárható, hogy legalább két hasznos, és két felismert ajtó vonalat találjon az algoritmus. A színes képen Canny élkiemeléssel lehet a legtöbb vonalat kinyerni, Megállapítható, hogy a színes képen futtatott algoritmusok nagyobb bizonytalansággal ismerik fel az ajtóéleket, szemben a mélységképen futatottakkal, amelyek a diagram szerint minden képen ugyanannyi hasznos vonalat találtak.



13. ábra A különböző logikai feltételeknek megfelelő vonalak száma algoritmusomként

A logikai feltételeknél nehéz lenne meghatározni az ideális határt, ugyanis a különböző algoritmusok egyike kevés, de helyesen felismert élből is helyesen találja meg az ajtót, másik módszer, például a színes képen alkalmazott Canny élkiemelő sokkal több, a 16. ábra alapján átlagosan 6 hasznos vonalat ismer fel, de végeredményben mindkettő egyaránt jól teljesít. Az ábrán látható, hogy az ÁM feltététel szűri meg legjobban a vonalpárokat, ugyanis annak a feltételnek felel meg a legkevesebb, viszont hibásan felismert ajtó a tesztképeken nem volt, így amely ennek a feltételnek megfelel az ténylegesen egy ajtót jelöl. Mivel a 3 tesztképen összesen 3 ajtó volt így minimum 3 olyan vonalpárnak kellene lennie, amely minden feltételnek megfelel. Ez a határ látható a 17. ábrán fekete vonallal. A „+” jellel jelölt oszlopok a jelmagyarázatban található algoritmusok együttes használatával elért eredményt tartalmazzák.



14. ábra A felismert lehetséges ajtók száma algoritmusoként

A 18. ábra azt mutatja, hogy a színes képen Canny élkereséssel és mélységképen Laplace, valamint Differenciális élkereséssel ismerhető fel a legtöbb ajtó. Fontos megjegyezni, hogy egy képen, amelyen egy ajtó található, az algoritmus több „lehetséges ajtót” is találhat, például, ha színes és mélységképen is keresünk egyszerre, mivel akkor a mélységkép eltolja és torzítja a színes képen található éleket. Azok az algoritmusok, amelyek 1 feletti értéket mutatnak, jól teljesítettek, mivel minden képen átlagosan egy ajtót felismertek. Ezt a határt szintén fekete vonallal jelöltem.

H. Konklúzió

Az „SZ, C+M, L” valamint az „Sz, C+M, L” jelzésű algoritmus kombinációk érték el a legjobb eredményt a tesztképeken. A mérési eredményekből látható, hogy a színes képen felismert vonalak eltérése kisebb, viszont erősen függ a megvilágítástól és a fényerőtől, valamint az árnyékokat is vonalnak ismerheti fel. A mélységkép nem függ ezektől, viszont sokkal zajosabb a kép. A felismerő algoritmus tárgyak olyan konstellációját, amely, az ajtóhoz hasonló geometriai tulajdonságokat hordoz, felismerheti ajtóként. Például, ha két szekrény kellő távolságra vannak egymástól, és a köztük átjárható hely van. Ezt a problémát jelen dolgozatban nem vizsgáltam.

KÖSZÖNETNYILVÁNÍTÁS

A publikáció elkészítését az EFOP-3.6.1-16-2016-00022 számú projekt támogatta. A projekt az Európai Unió

DOI: 10.17667/riim.2018.si/17.

támogatásával, az Európai Szociális Alap társfinanszírozásával valósult meg.